



UNIVERSITY OF SURREY

EEE3027: Final Essay

16th of May 2023

URN: 6596386

School of Computer Science and Electronic Engineering
Faculty of Engineering and Physical Sciences
University of Surrey

Contents

A System on a Chip Interfacing	1
A)a JTAG Advantages	1
A)b JTAG Functionality	1
A)c VHDL Implementation	2
A)d Calculations Single Stage	2
A)e Calculations Four Core	2
B Design Challenge	3
B)a Architecture and Block's Design	3
B)b VHDL Implementation	3
I References	4

A System on a Chip Interfacing

A)a JTAG Advantages

As chips have become denser, directly probing pins, such as with the old bed-of-nails technique, is no longer possible, yet accessing these pins is vital for testing. This is the problem the Joint Test Action Group set out to solve with JTAG, which is a specification for accessing on-chip resources and boundary-scan hardware testing on the board[1]. Without boundary-scanning, the only test option is functional, which requires massive amounts of test cases to exhaust all possibilities and does not help find the cause of a failure. Meanwhile, JTAG's boundary scan enables one to control and monitor individual IO pins, allowing for structural tests. Structural tests will help find assembly defects, such as shorts, and run much quicker as they are non-exhaustive.

JTAG also offers other benefits, such as monitoring systems during operation without disrupting them and even testing systems that are otherwise non-functional[2]. This is because JTAG is independent of the system and accessible via just four pins (depending on specification). As the logic and pins are part of the IC, JTAG will allow for testing throughout a product's life cycle. Engineers can use the interface during prototyping, production, and after deployment[2]. JTAG also daisy chains, allowing multiple devices within a system to be tested with one interface.

(208)

A)b JTAG Functionality

The JTAG implementation shown in Figure 1 of the essay brief connects to the internal bus of the ARM Cortex-M0 SoC. This connection enables the JTAG interface to control and monitor the data on that bus, allowing one to shift in a test pattern to test the NVIC, processor core, debug subsystem, memory and peripherals, or any combination thereof.

This sort of boundary testing is done via a path of interconnected shift-register cells on each signal pin in the bus, which are controlled via the Test Access Port (TAP) of the JTAG. The TAP has four core control signals[3]:

- TCK, clock to synchronize operation
- TMS, mode select line
- TDI, input shifted into cells on the rising edge
- TDO, output shifted out of cells on the falling edge
- (Optional) TRST, a reset for the state machine

Creating this functionality in VHDL is done by creating a state machine, shown in Figure 1, which is controlled by the value of TMS during the rising edge of TCK. The state machine has two paths to capture and control the data on the instruction registers (IR) or the data registers (DR).

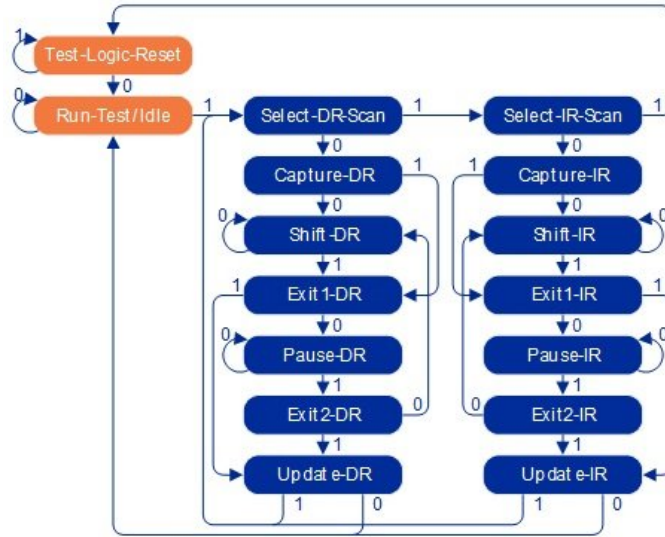


Figure 1: JTAG State Machine Diagram from Corelis[2]

The cells must also be created in VHDL using multiplexers and latches. These cells have a parallel input and output, which allow for the regular operation of the device, and shift inputs and outputs, which are used during tests to load data in or read data out.

(235)

A)c VHDL Implementation

CLUELESS

A)d Calculations Single Stage

One can calculate throughput, latency, and clock period given time per unit of work and the register overhead. Throughput (ω) is the number of tasks a module can solve in a given period of time, with a standard unit of mega-operations (millions of operations) per second or mops[4]. Latency (T) is the time taken per task[4]. These two values are reciprocal ($\omega = \frac{1}{T}$). The clock period (T_{clk}) is the time from one rising edge to the next

For a generic single-stage processor, which takes 24 ns per instruction (T_m), and with flip-flops at the input and output, with 300 ps overhead(t_{reg}), the following values can be calculated:

$$T = T_m + t_{\text{reg}} = 24 \text{ ns} + .3 \text{ ns} = 24.3 \text{ ns} \quad (1)$$

$$\omega = \frac{1}{T} = \frac{1}{24.3} = 41.15 \text{ Mops} \quad (2)$$

$$T_{\text{clk}} = 24.3 \text{ ns} \quad (3)$$

This system area usage could also be calculated if the area of the module (a_m) and the area of the register (a_{ref}) were known, using the following equation.

$$a = a_m + a_{\text{ref}} \quad (4)$$

(133)

A)e Calculations Four Core

Expanding on part A)d with a four-core system with nine stages, the calculations differ slightly.

Firstly the effect of using nine stages ($n = 9$) is calculated, as there is now more register overhead[4] (T_m and t_{reg} are unchanged).

$$T = T_m + nt_{\text{reg}} = 24 \text{ ns} + 9 * .3 \text{ ns} = 26.7 \text{ ns} \quad (5)$$

$$\omega_{\text{pipeline}} = \frac{1}{T_m/n + t_{\text{reg}}} = \frac{1}{24 \text{ ns}/9 + .3 \text{ ns}} = \frac{1}{2.967 \text{ ns}} = 337.08 \text{ Mops} \quad (6)$$

$$T_{\text{clk}} = 26.7 \text{ ns}/9 = 2.967 \text{ ns} \quad (7)$$

The four-core nature of the new system does not add any timing overhead but will increase throughput four-fold, as the task will be completed in parallel.

$$\omega = \omega_{\text{pipeline}} * 4 = 337.08 \text{ Mops} * 4 = 1348.31 \text{ Mops} \quad (8)$$

These changes make the new system 32 times faster than the one in part A)d. The cost of this new design's performance is a considerable increase in area. The new requirement can be calculated as follows, with n being the number of stages and p the number of parallel systems:

$$a = p(a_m + na_{\text{ref}}) \quad (9)$$

$$a = 4a_m + 36a_{\text{ref}} \quad (10)$$

$$(11)$$

Pipelining must also be done in a manner that avoids bottlenecking the system, as the slowest module limits throughput.

(135)

B Design Challenge

B)a Architecture and Block's Design

AMBA, or Advanced Microcontroller Bus Architecture, is an open Architecture used as the standard solution for how blocks on an SoC or ASIC interface[5]. The Cortex-M design uses the AHB (High-performance Bus) protocol. SRAM, static random access memory, is a type of volatile memory (data is only retained when powered) [4]. SRAM is most easily compared to DRAM, with the key difference being structure. SRAM consists of flip-flops requiring multiple transistors, which incurs a higher cost per GB when compared to DRAM. However, using flip-flops means SRAM need not be refreshed the way DRAM must, allowing SRAM to offer quick (single cycle) memory off-chip while also being lower power. As size per chip is limited, multiple SRAM chips are often used together.

As with most memory SRAM requires an address, data input, and write signal and produces only a data output. Most SRAM is also synchronous, requiring a clock.

Hamming codes are [6] [7]

B)b VHDL Implementation

I References

- [1] L. Pittroff, *Tutorial: The role of jtag in system debug and test throughout the embedded system development lifecycle*, Oct. 2008. [Online]. Available: <https://www.embedded.com/tutorial-the-role-of-jtag-in-system-debug-test-throughout-the-embedded-system-development-lifecycle/> (visited on 05/15/2023).
- [2] *Jtag test overview*, Nov. 2022. [Online]. Available: <https://www.corelis.com/education/tutorials/jtag-tutorial/jtag-test-overview/> (visited on 05/15/2023).
- [3] R. Oshana, *Jtag: An introduction*, Jan. 2023. [Online]. Available: <https://www.embedded.com/introduction-to-jtag/> (visited on 05/15/2023).
- [4] W. J. Dally, R. C. Harting, and T. M. Aamodt, *Digital Design using VHDL A Systems Approach*. Cambridge University Press, 2016.
- [5] B. Walshe, *What is amba?* Dec. 2014. [Online]. Available: <https://community.arm.com/arm-community-blogs/b/architectures-and-processors-blog/posts/what-is-amba> (visited on 05/15/2023).
- [6] J. R. Barry, *Digital communication*. eng, 3rd ed. / John R. Barry, Edward A. Lee, David G. Messerschmitt. Boston, Mass. ; Kluwer Academic, 2004, ISBN: 0792375483.
- [7] U. Sani and I. Shanono, "Design of (7, 4) hamming encoder and decoder using vhdl,," Sep. 2015.